Introduction to Port Extension

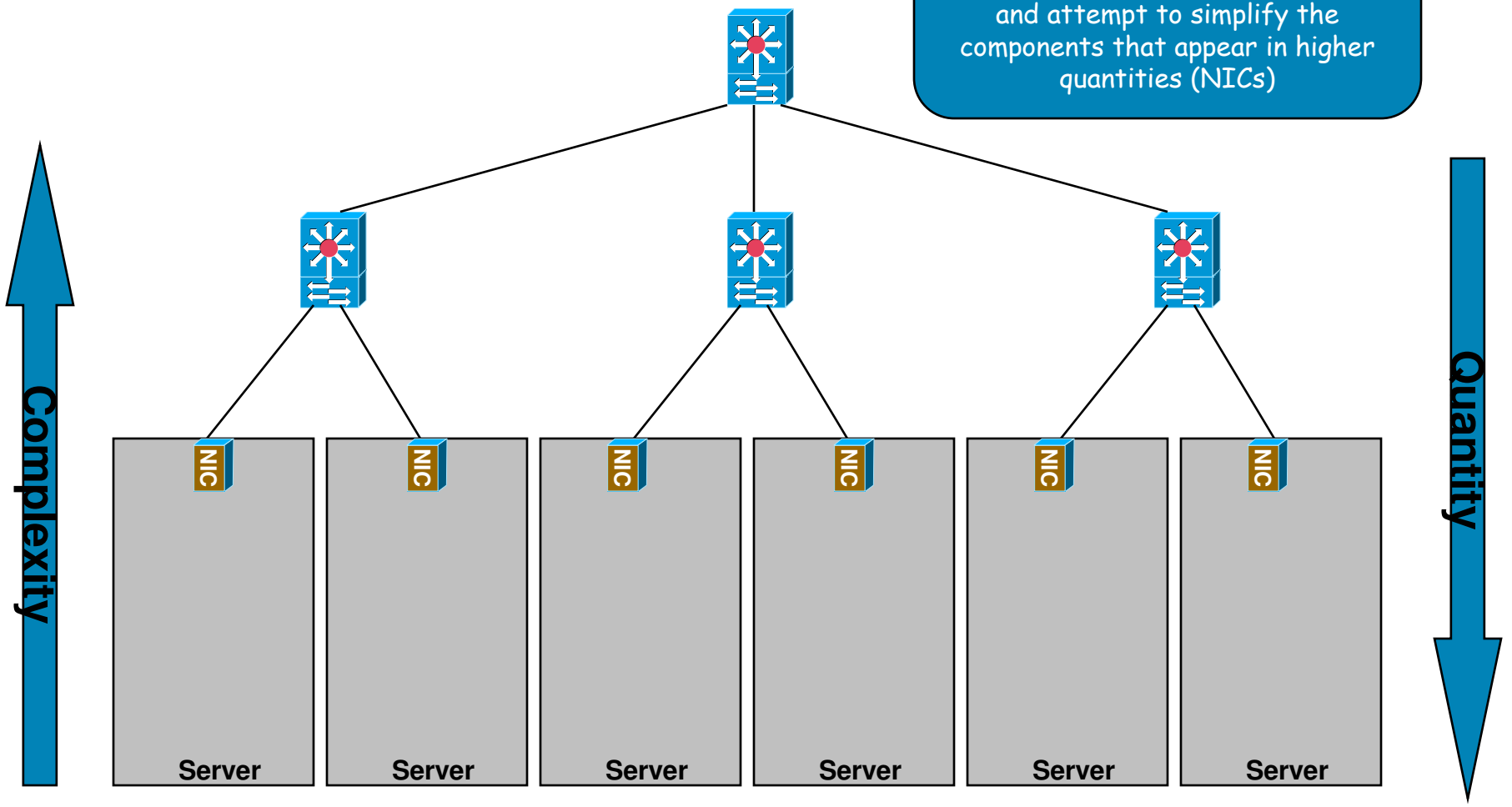
Joe Pelissier
jopeliss@cisco.com

Agenda

- Motivation, Problem Statement, and Requirements
- An Approach
- The Port Extension Proposal
- The Path to Tagging
- Summary

Motivation

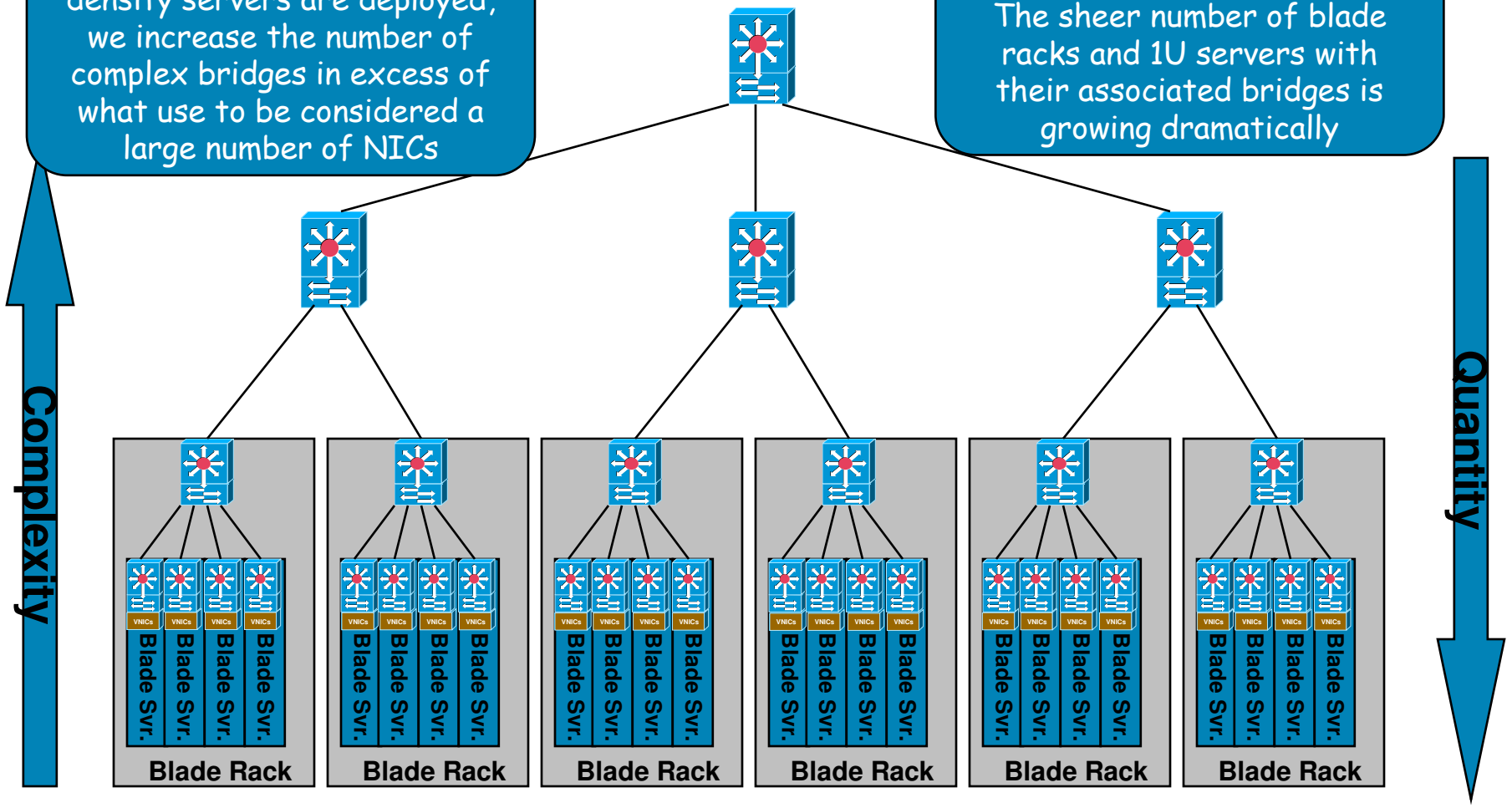
As a general rule, we push complexity up into the components of which we have fewer (bridges), and attempt to simplify the components that appear in higher quantities (NICs)



Motivation

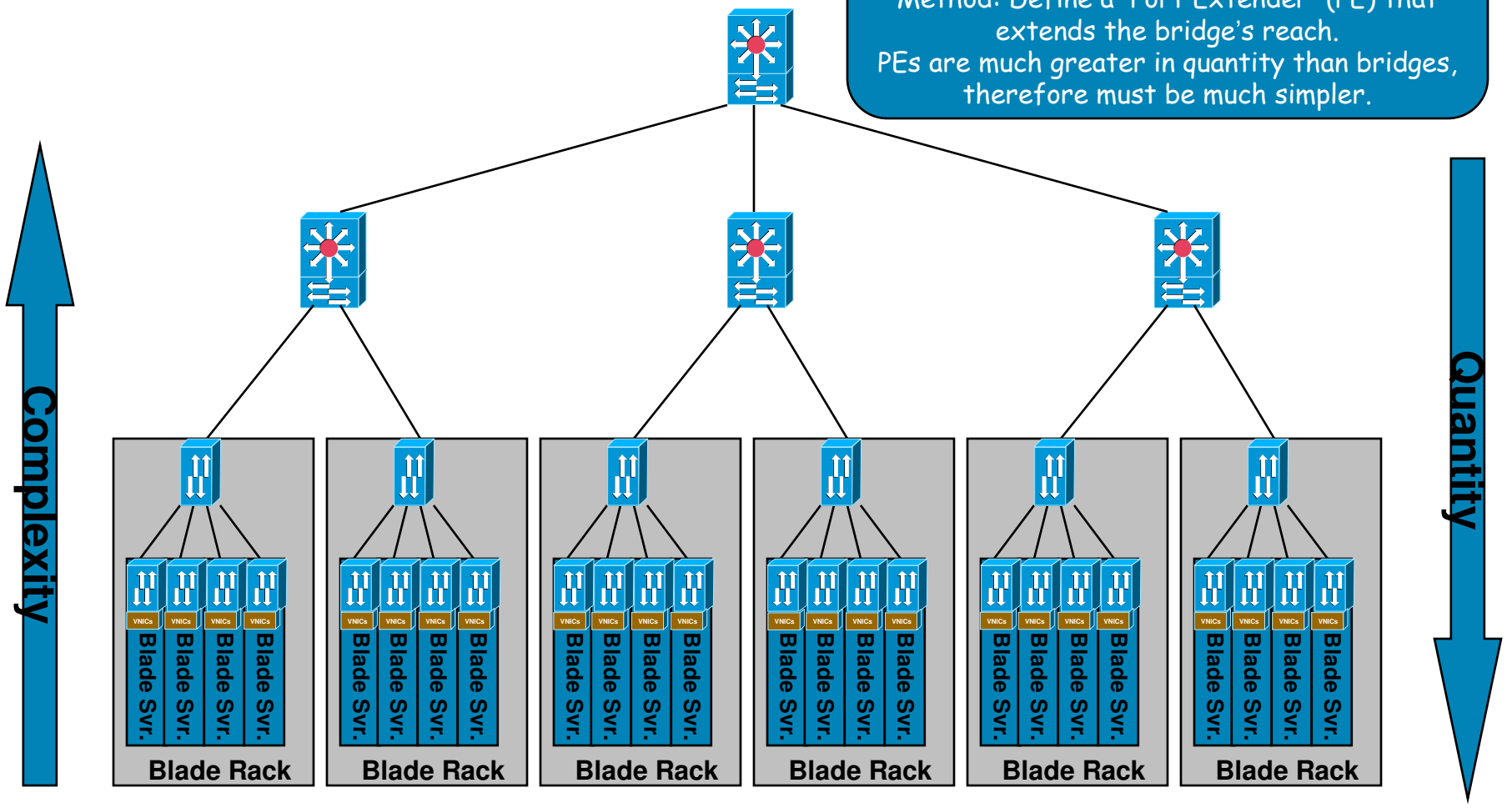
As virtualization and high density servers are deployed, we increase the number of complex bridges in excess of what use to be considered a large number of NICs

Even without virtualization, the same challenges exist. The sheer number of blade racks and 1U servers with their associated bridges is growing dramatically



Motivation

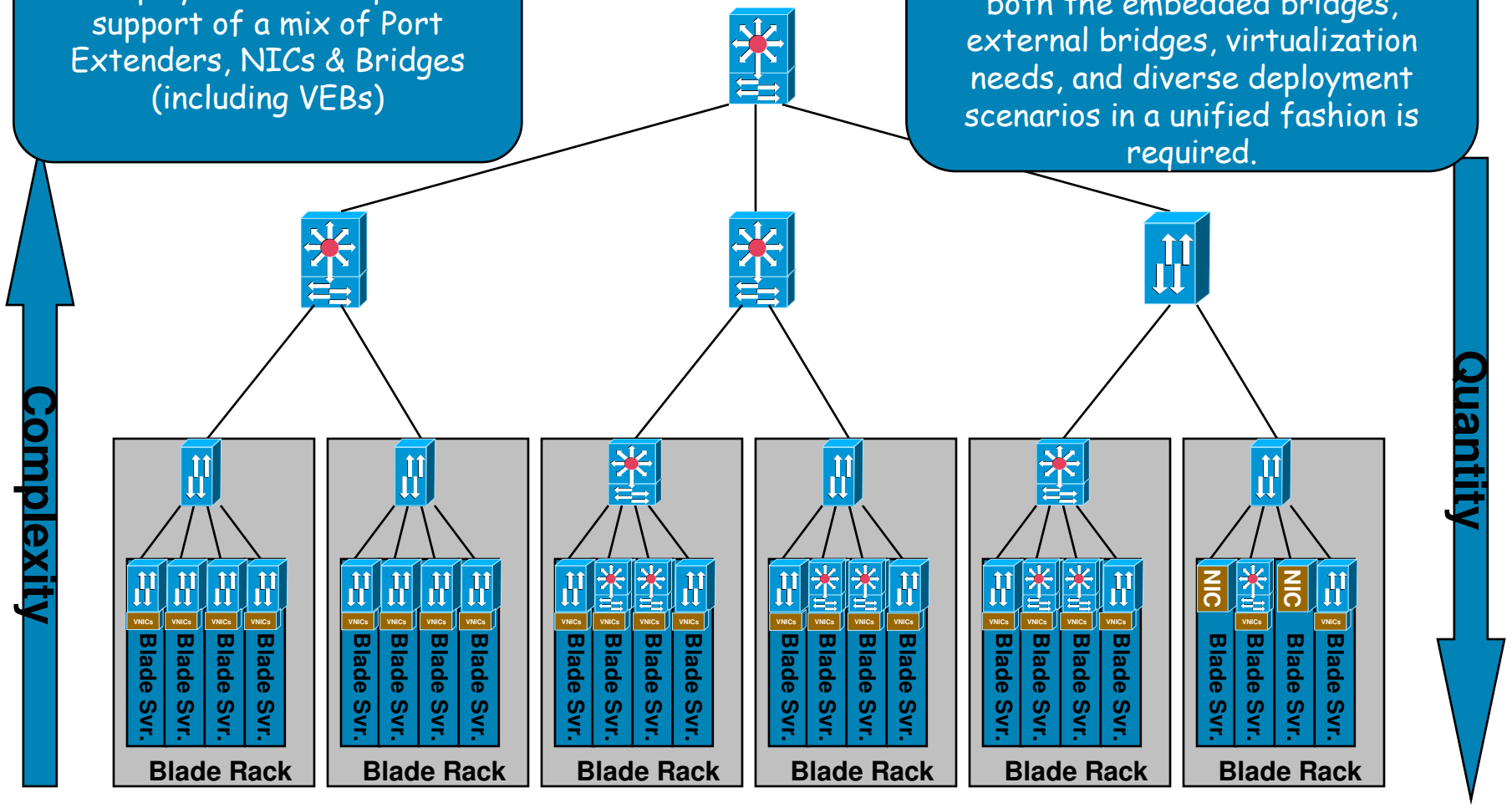
Goal: Extend the bridge into the blade racks and hypervisors, reducing the number of these complex devices.
Method: Define a "Port Extender" (PE) that extends the bridge's reach.
PEs are much greater in quantity than bridges, therefore must be much simpler.



Motivation

Deployment will require support of a mix of Port Extenders, NICs & Bridges (including VEBs)

It is insufficient to address only the "embedded" portion of the problem. A solution that addresses both the embedded bridges, external bridges, virtualization needs, and diverse deployment scenarios in a unified fashion is required.



Problem Statement

- **The deployment of hundreds to thousands of bridge devices with diverse capabilities and performance as a result of high density server technology (including but not limited to server virtualization) creates the following challenges that are addressed by the proposed technology:**
  - **High network management complexity and administrative cost**
  - **High initial capital expenditures**
  - **Stressed scalability limits and responsiveness of network management applications due to:**
    - Volume of points of management
    - Volume of management messages required
- **Addressing just the embedded bridge in virtualized servers is insufficient to address the overall problem**
  - **Both embedded and external bridges contribute to the problems**

Requirements Summary

- **Must provide the same behavior to the station (i.e. NIC or VNIC) that is provided today by bridges**
  - **Fundamental to interoperability**
  - **Deviating from such behavior opens the door for unforeseen consequences**
  - **Extremely undesirable to require applications to be aware of whether they are directly connected to a bridge versus a PE).**

Requirements Summary

- **Must be simple**
  - **Drive complexity towards the bridge and simplicity towards the NIC**
    - For example, ACL processing, CAM lookups, learning and aging functions, etc.
  - **Complexity should be limited to fewer devices**
    - Simplifies management
    - Lowers TCO
    - Simplifies upgrades
    - Etc.
  - **Does not eliminate the need for embedded bridges**
    - Embedded bridges optimal for high bandwidth VM to VM communication
    - Port Extension optimal for high bandwidth VM to network communication
    - Simplicity provides the differentiation for use in the appropriate segments

Requirements Summary

- **Must efficiently support embedded bridging**
- **Must efficiently support converged networking**

  These technologies expect certain functions commonly available in bridges today

  VLAN enforcement, locally assigned MAC addresses, basic ACL capabilities, static forwarding entries, etc.

  Must ensure these capabilities carry forward

Requirements Summary

- **Must provide simple and efficient management capabilities**
  - **Reducing "points of management" is a good start**
    - However, if a "point of management" must initiate additional management messages, little has been gained
  - **Must provide predictable and consistent capabilities**
    - e.g. reduce fabric dependencies for VM migration and converged networking
  - **Reduce the number of devices that are "touched" by a management operation**
- **Must be cost effective**
  - **Otherwise there is no point…**
  - **The cost vs. benefit must be superior to other approaches**
- **Must minimize changes to bridge architecture**
  - **No need for invention for its own sake**
  - **Reuse proven technology and methods**

An Approach

An Approach

- **Port Extension proposes to meet the previously stated requirements by providing a capability to combine distributed network components into a single logical 802.1Q compliant bridge**
- **These components consist of:**

  A centralized Controlling Bridge

  Distributed Port Extenders (that may be cascaded)

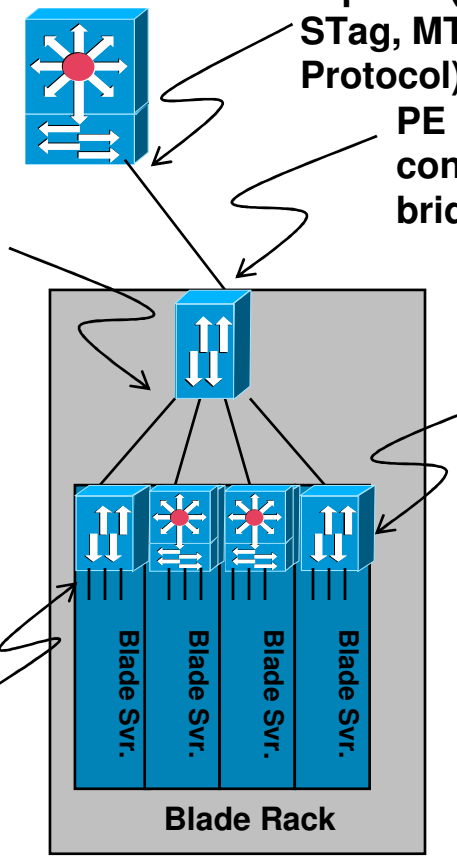
  A protocol enabling control of the Port Extenders by the Controlling Bridge
- **The set of the Controlling Bridge and the Port Extenders form a *single* 802.1Q compliant bridge**

An Approach - Anatomy of a Port Extended Network

Bridges that connect to PE Uplink Ports must be PE capable (e.g. support the STag, MTag, and the PE Protocol).

PE Uplink Port: may connect to an PE capable bridge or an PE downlink

PE Downlink Port: may connect to an PE Uplink Port, a bridge or VEB, or a NIC (virtual or physical). Note that the bridge does not need to be PE capable in this case.



PEs may be cascaded. In this case, the Downlink Ports (virtual in this example) act as ports of the top level bridge.

Downlink ports are assigned a PE Port Identifier (PepID) that corresponds to an interface on the Controlling Bridge and is used to route frames down through PEs

Note: multiple Uplink Ports connected to different bridges or PEs are supported.

An Approach - Observations

- **To the greatest extent possible, all bridging functions are performed in the Controlling Bridge**
  - **Many bridging functions require knowledge of the ingress and/or egress port. A tag provides this information**
- **The ports on the south side of an PE are *physical* ports**
  - **You can see, touch, smell, and taste them**
  - **If you plug a network analyzer into one, it will see an 802.1Q compliant bridge**
    - The tags are limited between the PE and Controlling Bridge, so you would never see one at this point
  - **Inserting an PE is similar to inserting a line card**
    - New ports are instantiated in the Controlling Bridge just as if a line card was inserted
    - These ports are managed just as if they were part of a new line card
  - **There is nothing virtual about it!**
- **The ports of an embedded PE may be "virtual"**
  - **That is, they are conceptual and connect to a conceptual NIC (commonly referred to as a virtual NIC).**
  - **However, from the point of view of the Controlling Bridge and management of these ports, they are handled just like any other port**

The Port Extension Proposal

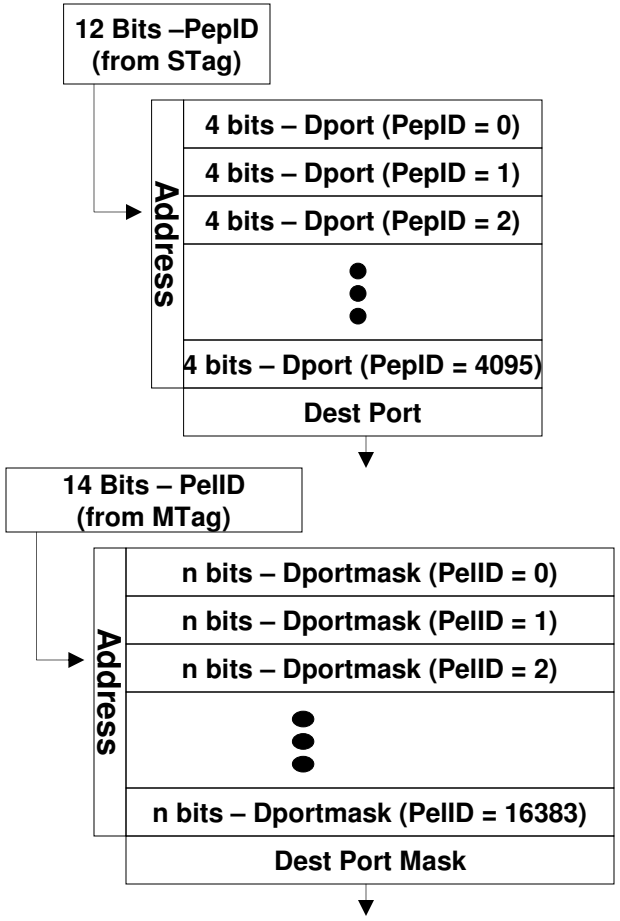
PE Downlinks & PE Port Identifiers

- **Each downlink from an PE to a NIC, VNIC, bridge, or VEB is, in effect, a bridge interface**
  - **These are the instantiations of interfaces of the Controlling Bridge**
  - **Each downlink identified by a 12-bit PE Port ID (PepID)**
    - Assigned by the bridge to each PE downlink port at PE initialization
    - Scope of uniqueness is the Controlling Bridge Port
- **PepID is carried in the SVID field of an STag**
  - **STags are currently defined in IEEE specifications for use in provider bridging**
  - **Communicating PepID in an STag is a new use of the existing tag**

Port Extender List Identifiers (PeIIDs)

- 14-bit identifier used for flooding and multicast
- Identifies a group of ports to which a frame is to be replicated
- Port Extender egress ports responsible for blocking a replicated frame if the egress port was also the ingress port
  - Source port PepID provided for this purpose
- PeIIDs are communicated using a new tag called the “MTag”
  - The MTag also carries the source PepID as mentioned above

PE Forwarding Tables



- **PE forwarding table (used for unicast)**
  - **One entry per PepID**
    - May support up to 4096 unique PepIDs
    - Indexed by DPepID (the SVID field of the STag)
    - Each entry points to the downlink to be used
- **PE list table (used for multicast and flooding)**
  - **One entry per PelID**
    - May support up to 16k unique lists
    - Indexed by PelID (from the MTag)
    - Each entry contains a bit mask indicating which downlinks are to be used
      - Width of entry depends on number of downlink ports
- **Note: Table size not a function of VLANs / MAC addresses in use**
  - Each interface utilizes a single entry regardless of the number of VLANs and / or MAC addresses in use on that interface

Port Extender Basic Functions

- **From NIC to Controlling Bridge**
  - **Add STag if none present (indicating source PepID)**
    - STag added *only* at ingress
      - STags are not “stacked” as the frame passes through successive PEs
  - **Forward frame up the PE hierarchy to the Controlling Bridge**
- **From Controlling Bridge to NIC**
  - **Froward frame down hierarchy to the NIC**
    - Destination port determined by using DPepID from STag as index into the forwarding table (unicast) or PelID from MTag (multicast)
  - **Replicate multicast frames**
    - Filter the frame at the ingress port if it was sourced at the PE
      - (i.e. if the port’s assigned PepID matches the SPepID in the MTag)
  - **Remove the STag / MTag if the final downlink has been reached**

Bridge use of STag and MTag

- **On ingress**

  Learn source PepID along with MAC address, VID, and port number as part of normal bridge learning function

- **Forwarding**

  Utilize source PepID along with ingress port number as frame source for all normal bridge functions (ACLs, VLAN member set enforcement, etc.)

- **On egress:**

  Populate the STag with the destination PepID (unicast) or the MTag with the source PepID and destination PelID (multicast)

Support of Bridge and other PDUs

- **The set of a Controlling Bridge and its Port Extenders form an 802.1Q compliant bridge**

  Implies that the Controlling Bridge must have the ability to send arbitrarily addressed protocol frames to specific PE egress ports (e.g. BPDUs), and to identify from which port these frames were received (independent of source MAC address)

  Solution:

  For transmission, address the protocol frame as appropriate, and direct it to the desired PE egress port with an appropriate STag

  On reception, the STag provides the identity of the port from which the frame was received

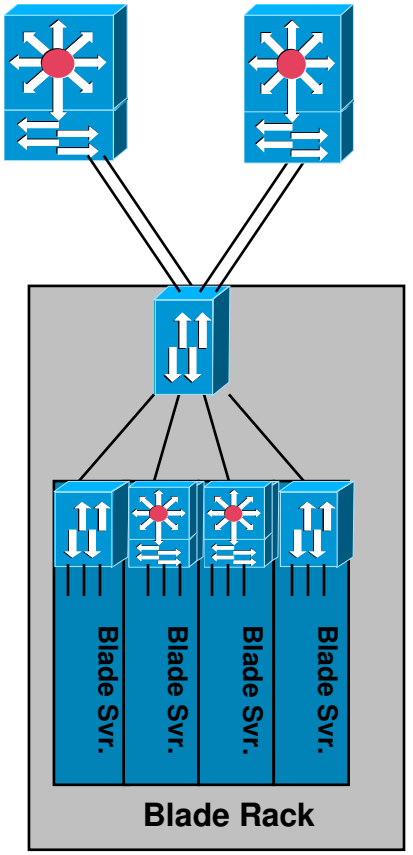
- **Note: this is the same function that is performed internal in bridges today**

  Every frame received by the bridge's control processor is somehow marked with an ingress port number indication

  Every frame transmitted by the bridge's control processor is somehow marked with an egress port number indication

Support of Multiple Uplink Ports

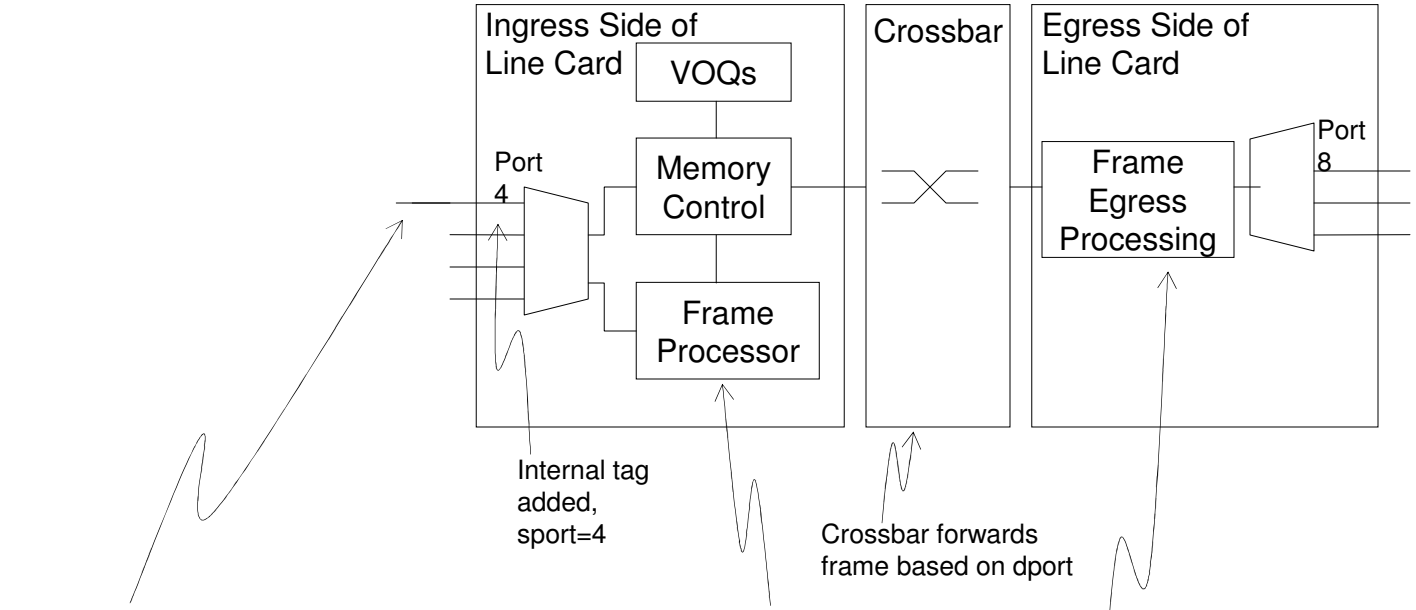
- **Required for:**
  - **Redundancy**
  - **Support of multiple fabric connectivity**
- **Achieved by:**
  - **Instantiating a forwarding table and list table for each uplink port**
    - Addresses “Southbound” frames
  - **Each downlink port is associated with a single uplink port**
    - All frames received on that downlink port are forwarded to the associated uplink port
    - Addresses “Northbound” frames



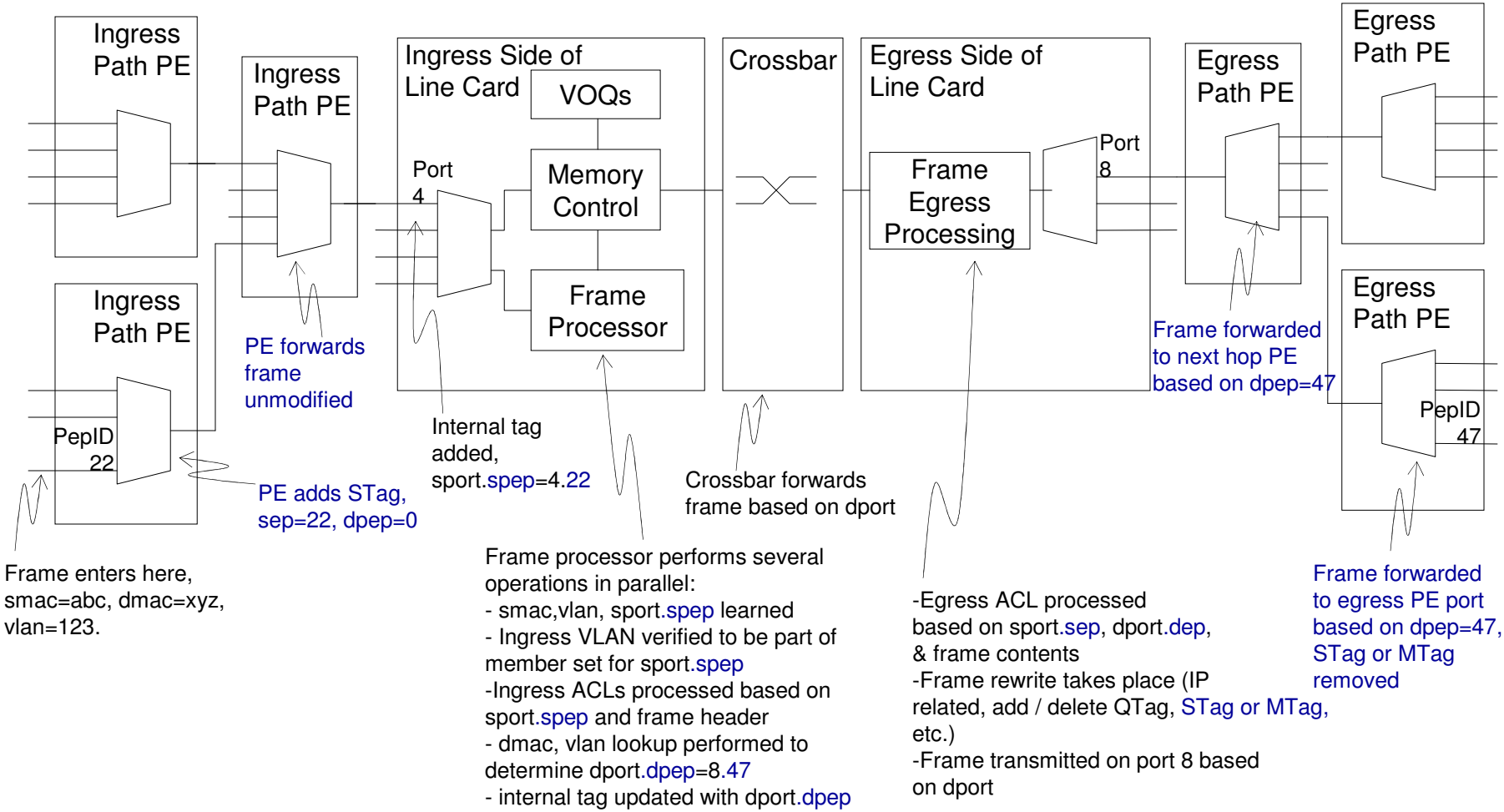
The Path to Tagging

Tagging is a natural extension of Bridge Functionality

The Path to Tagging



The Path to Tagging



The Path to Tagging - Observations

- When a frame enters a bridge, it is internally "tagged" with an indication of the ingress port
- The ingress port is used in several frame processing operations, ultimately resulting in determination of the egress port, which is added to the internal tag
- The rest of the forwarding through the bridge is performed based on the internal tag
- At egress, egress ACL processing is performed based on ingress port, egress port, and Frame Contents *(on a per egress port basis for multicast).* Frame processing adds or removes a QTag, and potentially other packet rewrite functions
- With Port Extension, all of the fundamental bridge functionality remains identical
  - Which is a very good thing ☺
  - From the outside world, the combination of PEs and the controlling bridge is a single 802.1Q compliant bridge
- The PEs are extremely simple
  - On ingress, add a tag, then forward north
  - Southbound, forward based on ep_id as index into forwarding table
    - Remove STag or MTag at a last hop

Summary

Summary

- **Port Extension provides a straight forward approach to address the problems associated with bridge proliferation in modern data center environments**
  - **Provides a simple, low-cost alternative that can dramatically reduce the number of bridges**
  - **Interoperates with and complements independent bridges, including VEBs and new Data Center Technologies under development such as VEPA**
  - **Addresses data center "pain points" beyond just the bridge embedded in a virtualized server in a logical and consistent manner**

Questions?

Thank You!